

GENERATIVE ADVERSARIAL NETWORKS

Marek Dobransky

17/05/2019

Papers

- [1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio: Generative Adversarial Networks
- [2] Martin Arjovsky, Soumith Chintala, Léon Bottou: Wasserstein GAN
- [3] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville: Improved Training of Wasserstein GANs

Introduction

In this summary we will introduce three paper. The first one [1] introduces the notion of Generative adversarial networks and proposes a training process based on min-max game between generator and discriminator. The other one [2] identifies the problems with convergence of the approach and proposes a use of Wasserstein metrics to optimize the discriminator. The final paper we reviewed [3] continues the process by identifying the shortcomings of [2] and proposes an adjustment to the loss function.

[1] Generative Adversarial Networks

In the adversarial framework, the generative model is pitted against an adversary: a discriminative model that learns to determine whether a sample is from the model distribution or the data distribution. In this article, the authors explore the special case when the generative model generates samples by passing random noise through a multilayer perceptron, and the discriminative model is also a multilayer perceptron. Both models are trained by backpropagation and samples from the generative model are created by forward propagation.

Definition

The generator, discriminator and corresponding data distributions are defined as follows:

- $G(z; \theta_g)$ is a multilayer perceptron with parameters θ_g that maps input noise z into a data space.
- $D(x; \theta_d)$ is a multilayer perceptron with parameters θ_d that outputs a probability that x comes from the data rather than generator.
- p_z is a data distribution over noise input z
- p_g is a generator's distribution over data x
- p_{data} is a data distribution over real sample x

The discriminator is trained to produce a high probability given a real sample and a probability close to zero given a fake sample. On the other hand, generator is simultaneously trained to increase

the chance of discriminator producing a high probability given a fake sample. This represents a minmax game between D and G with value function $V(D, G)$:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Training

The training process then follows the following algorithm:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k , is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Sample minibatch of m examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)}))) \right].$$

end for

- Sample minibatch of m noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)}))).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Theory

The paper proceeds to theoretically prove that the optimum is reached when $p_g = p_{data}$, and convergence of the algorithm to this optimum.

Most importantly, it shows that minimizing the optimal discriminator loss, with respect to the generator model parameters is equivalent to minimizing Jensen–Shannon divergence $JS(p_{data}||p_g)$.

Experiments

The paper provides a few examples of this approach using a image generation. Here are two examples on fig. 1 and fig. 2.



Figure 1: Digits obtained by linearly interpolating between coordinates in z space of the full model.

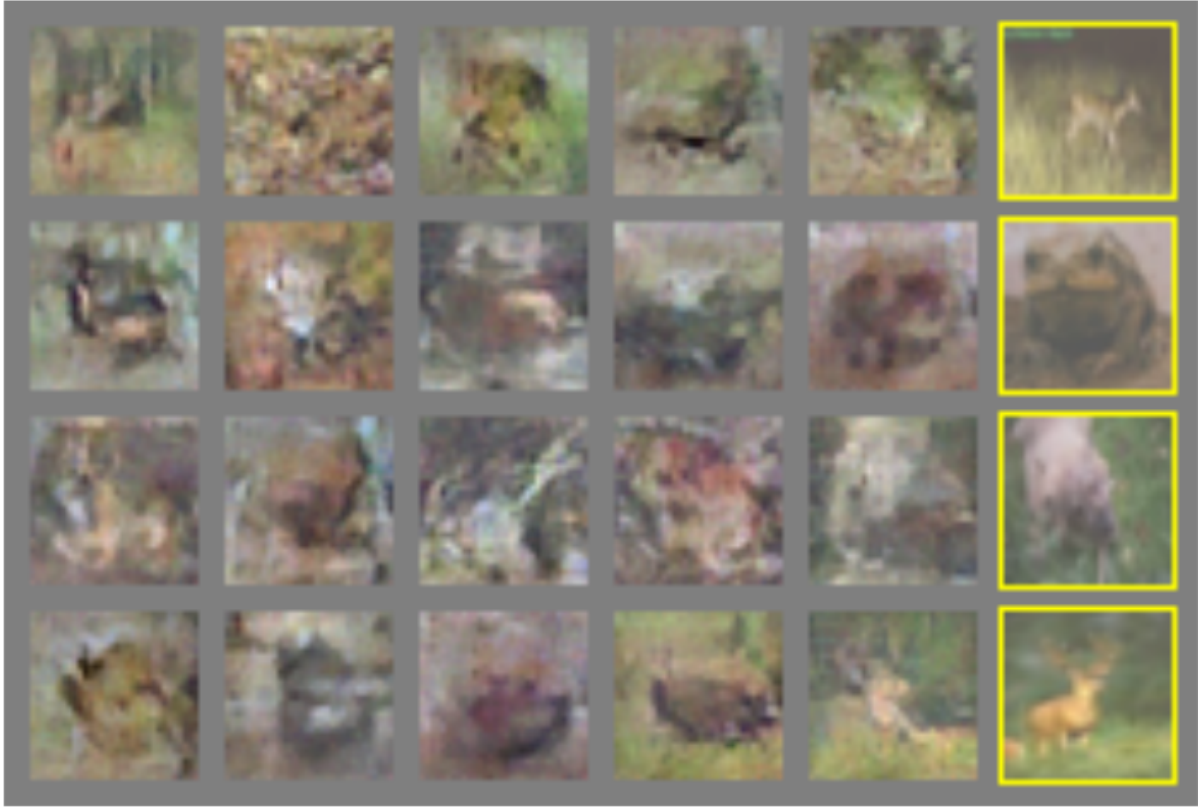


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. CIFAR-10 example with convolutional discriminator and deconvolutional generator.

[2] Wasserstein GAN

This paper begins by talking about generative models and learning of probability distribution. It provides a comparison of Earth Mover distance (or Wasserstein) with other probability distances and divergences and defines a form of GAN called Wasserstein-GAN that minimizes a reasonable and efficient approximation of the EM distance, and theoretically shows that the corresponding optimization problem is sound.

Probability distribution distances

Definitions of multiple distances between probability distributions \mathbb{P}_r and \mathbb{P}_g :

The Total Variation(TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

The Kullback-Leibler(KL) divergence

$$KL(\mathbb{P}_r \parallel \mathbb{P}_g) = \int \log \left(\frac{P_r(x)}{P_g(x)} \right) P_r(x) d\mu(x)$$

where both \mathbb{P}_r and \mathbb{P}_g are assumed to be absolutely continuous

The Jensen-Shannon(JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r \| \mathbb{P}_m) + KL(\mathbb{P}_g \| \mathbb{P}_m)$$

where \mathbb{P}_m is $(\mathbb{P}_r + \mathbb{P}_g)/2$

The Earth-Mover(EM) distance or Wasserstein

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively \mathbb{P}_r and \mathbb{P}_g

Example

The paper introduces a simple example to argue why we should care about the Earth-Mover distance.

Let $Z \sim U[0, 1]$ the uniform distribution on the unit interval. Let \mathbb{P}_0 be the distribution of $(0, Z) \in \mathbb{R}^2$, uniform on a straight vertical line passing through the origin. Now let $g_\theta(z) = (\theta, z)$ with θ a single real parameter.

When $\theta_t \rightarrow 0$, the sequence $(P_{\theta_t})_{t \in \mathbb{N}}$ converges to P_0 under the EM distance, but does not converge at all under either the JS, KL, reverse KL, or TV divergences. The following figure illustrates this for the case of the EM and JS distances.

The authors conclude that the KL, JS, and TV distances are not sensible cost functions when learning distributions supported by low dimensional manifolds. However, the EM distance is sensible in that setup.

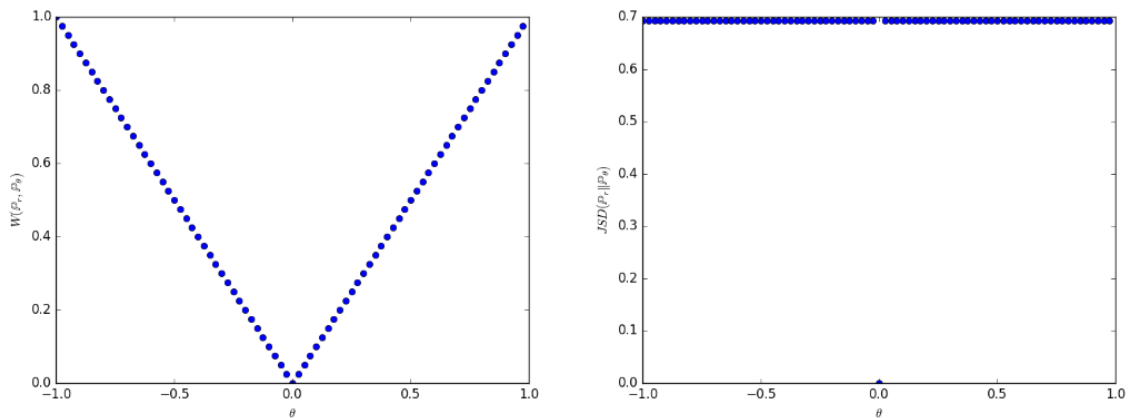


Figure 1: These plots show $\rho(\mathbb{P}_\theta, \mathbb{P}_0)$ as a function of θ when ρ is the EM distance (left plot) or the JS divergence (right plot). The EM plot is continuous and provides a usable gradient everywhere. The JS plot is not continuous and does not provide a usable gradient.

WGAN

Although the Wasserstein distance has some nicer properties than Jensen-Shannon for optimization, the infimum is highly intractable to compute. The authors use duality of Wasserstein metrics to simplify the calculation to:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} [f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta} [f(x)]$$

where the supremum is over all the 1-Lipschitz functions. Authors then propose the approximation of this function by using maximum instead of supremum and prove that it holds for K-Lipschitz functions.¹ $f : X \rightarrow R$.

Used in GANs, the optimized function f represents the critic function (discriminator in GANs) and is parametrized by the weights of the neural network. To satisfy the constrain for K-Lipschitz function, weights w are clamped to a compact space $[c, c]$.

The training process is described in the algorithm below.

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α , the learning rate. c , the clipping parameter. m , the batch size. n_{critic} , the number of iterations of the critic per generator iteration.

Require: : w_0 , initial critic parameters. θ_0 , initial generator's parameters.

```

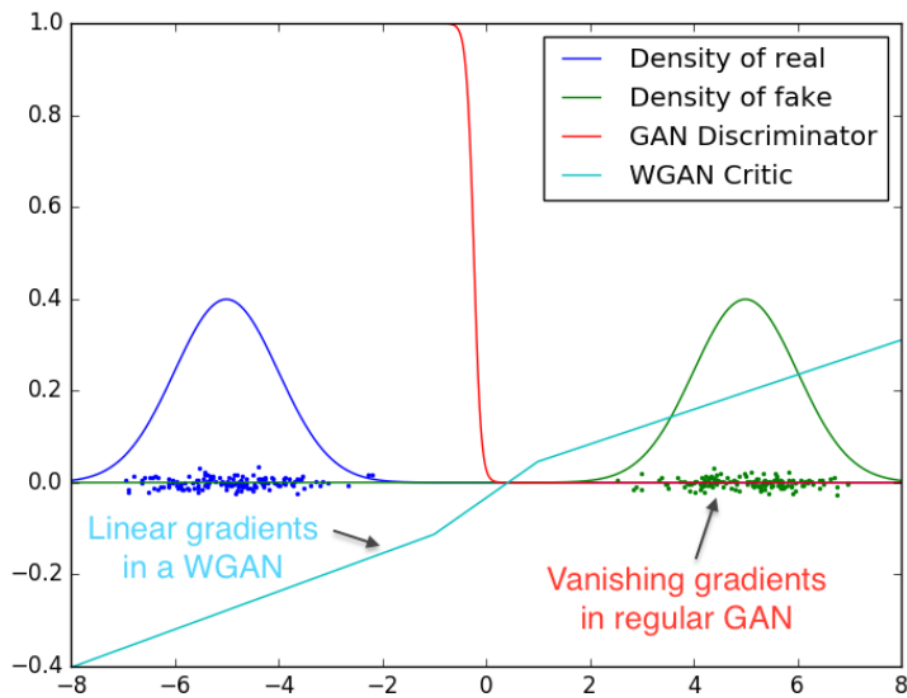
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w [\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while

```

Results

Optimal discriminator and critic when learning to differentiate two Gaussians. We can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Meanwhile WGAN critic provides very clean gradients on all parts of the space.

¹https://en.wikipedia.org/wiki/Lipschitz_continuity



The paper claims that benefits of WGAN are that it allows us to train the critic till optimality and removes the need to balance generator and discriminator's capacity properly. They observe that WGANs are much more robust than GANs when one varies the architectural choices for the generator, and provide a set of examples when WGAN learned successfully, whereas traditional GAN failed.

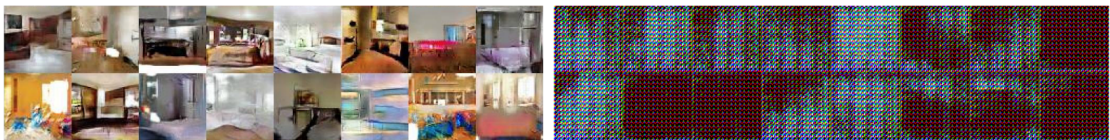


Figure 6: Algorithms trained with a generator without batch normalization and constant number of filters at every layer (as opposed to duplicating them every time as in [18]). Aside from taking out batch normalization, the number of parameters is therefore reduced by a bit more than an order of magnitude. Left: WGAN algorithm. Right: standard GAN formulation. As we can see the standard GAN failed to learn while the WGAN still was able to produce samples.

[3] Improved Training of Wasserstein GANs

This paper aim to improve the training of WGAN networks, and reduce the amount of cases where the training does not converge. The authors attribute those problem to the weight clipping in order to enforce Lipschitz constraint on the critic. This paper proposes an alternative to the weight clipping method.

Gradient penalty




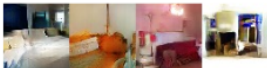

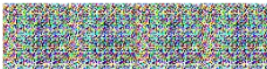



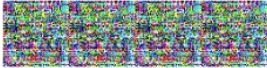




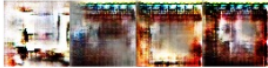













A differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. This method directly applies the norm constraint inside the lost function of the critic.

$$L = \underbrace{\mathbb{E}_{\tilde{x} \sim \mathbb{P}_g} [D(\tilde{x})] - \mathbb{E}_{x \sim \mathbb{P}_r} [D(x)]}_{\text{Original critic loss}} + \underbrace{\lambda \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]}_{\text{Our gradient penalty}}.$$

The $\mathbb{P}_{\hat{x}}$ samples data uniformly from the line between sampled points from data and generator distributions. The training also doesn't use batch normalization, as it would invalidate the training objective, that penalizes the norm of the critic's gradient with respect to each input independently.

Results

The authors show a set of experiments where other GAN architecture do not train well and WGAN-GP does.

DCGAN	LSGAN	WGAN (clipping)	WGAN-GP (ours)
Baseline (G : DCGAN, D : DCGAN)			
			
G : No BN and a constant number of filters, D : DCGAN			
			
G : 4-layer 512-dim ReLU MLP, D : DCGAN			
			
No normalization in either G or D			
			
Gated multiplicative nonlinearities everywhere in G and D			
			
tanh nonlinearities everywhere in G and D			
			
101-layer ResNet G and D			
			

My views

Likes

I like the way the paper continually improve the process and build on each other. I particularly like how the simple concept of two networks playing min-max game can lead to the generation of realistic images.

Dislikes

There is not much I don't like about the paper, except for the mathematical complexity, mainly in the WGAN approach. They expect a reader to have a in-depth knowledge of specific fields of mathematics (e.g. WGAN introduces Lipschitz functions without any explanation or reference).